Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет
по лабораторной работе №6

«Ансамбли моделей машинного обучения»

по курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:

Голубкова С.В.
Группа РТ5-61Б

_____

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

_____

Москва 2020

Лабораторная работа № 6

1. Цель лабораторной работы

Изучение ансамблей моделей машинного обучения.

2. Задание

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.

2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.

4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

3. Выполнение работы

Лабораторная работа №6 по курсу ТМО

"Ансамбли моделей машинного обучения".

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error, mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
```

1. Загрузка и обработка набора данных

Для решения задачи регрессии выберем набор данных Daily Temperature of Major Cities, который показывает среднесуточные значения температуры воздуха зафиксированы в крупных городах мира.

In [2]:
```
data = pd.read_csv('../datasets/city_temperature.csv')
data
```

```
c:\python_3.7.4\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (2) have mixed types.Specify dty
pe option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	Region	Country	State	City	Month	Day	Year	AvgTemperature
0	Africa	Algeria	NaN	Algiers	1	1	1995	64.2
1	Africa	Algeria	NaN	Algiers	1	2	1995	49.4
2	Africa	Algeria	NaN	Algiers	1	3	1995	48.8
3	Africa	Algeria	NaN	Algiers	1	4	1995	46.4
4	Africa	Algeria	NaN	Algiers	1	5	1995	47.9
...	...	...	...	...	...	...	...	...
2906322	North America	US	Additional Territories	San Juan Puerto Rico	7	27	2013	82.4
2906323	North America	US	Additional Territories	San Juan Puerto Rico	7	28	2013	81.6
2906324	North America	US	Additional Territories	San Juan Puerto Rico	7	29	2013	84.2
2906325	North America	US	Additional Territories	San Juan Puerto Rico	7	30	2013	83.8
2906326	North America	US	Additional Territories	San Juan Puerto Rico	7	31	2013	83.6

2906327 rows × 8 columns

Удаление пропусков в данных

In [3]:
```
data.isnull().sum()
```

Out[3]:
```
Region                0
Country               0
State           1450990
City                  0
Month                 0
Day                   0
Year                  0
AvgTemperature        0
dtype: int64
```

In [4]:
```
# удалим колонку state, тк штаты есть не во всех странах и нам этот прзнак не особо важен в понятийном смысле
data = data.dropna(axis=1, how='any')
data.shape
```

Out[4]: (2906327, 7)

Так как мы имеем очень много строк, то будем решать более узкую задачу и возьмем данные только за 2013 год:

In [5]:
```
data = data[data['Year']==2013]
data.shape
```

Out[5]: (111021, 7)

In [6]:
```
data = data.drop(['Year'], axis=1)
```

Кодирование категориальных признаков:

In [7]:
```
from sklearn.preprocessing import LabelEncoder
```

In [8]:
```
#Label encoding
le = LabelEncoder()
reg_enc_le = le.fit_transform(data['Region'])
country_enc_le = le.fit_transform(data['Country'])
city_enc_le = le.fit_transform(data['City'])
```

In [9]:
```
data_new = data.drop(['Region','Country','City','AvgTemperature'], axis=1)
data_new['Region'] = reg_enc_le
data_new['Country'] = country_enc_le
data_new['City'] = city_enc_le
data_new['AvgTemperature'] = data['AvgTemperature']
data_new
```

Out[9]:

	Month	Day	Region	Country	City	AvgTemperature
6575	1	1	0	1	7	49.2
6576	1	2	0	1	7	51.7
6577	1	3	0	1	7	48.2
6578	1	4	0	1	7	49.2
6579	1	5	0	1	7	49.4
...	...	...	...	...	...	...
2906322	7	27	5	103	244	82.4
2906323	7	28	5	103	244	81.6
2906324	7	29	5	103	244	84.2
2906325	7	30	5	103	244	83.8
2906326	7	31	5	103	244	83.6

111021 rows × 6 columns

2. Разделение выборки на обучающую и тестовую

In [10]:
```
X = data_new[['Month','Day','Region', 'Country', 'City']]
y = data_new['AvgTemperature']
print(X.head(), "\n")
print(y.head())
```

```
      Month  Day  Region  Country  City
6575      1    1       0        1     7
6576      1    2       0        1     7
6577      1    3       0        1     7
6578      1    4       0        1     7
6579      1    5       0        1     7

6575    49.2
6576    51.7
6577    48.2
6578    49.2
6579    49.4
Name: AvgTemperature, dtype: float64
```

Нормализация:

In [11]:
```
from sklearn.preprocessing import StandardScaler
```

In [12]:
```
# промасштабируем X
columns = X.columns
scaler = StandardScaler()
X = scaler.fit_transform(X)
pd.DataFrame(X, columns=columns).describe()
```

Out[12]:

	Month	Day	Region	Country	City
count	1.110210e+05	1.110210e+05	1.110210e+05	1.110210e+05	1.110210e+05
mean	6.912081e-18	5.056059e-17	2.048024e-16	-2.785313e-16	-1.454097e-16
std	1.000005e+00	1.000005e+00	1.000005e+00	1.000005e+00	1.000005e+00
min	-1.600388e+00	-1.673511e+00	-2.092386e+00	-2.252290e+00	-1.726359e+00
25%	-7.300681e-01	-8.777078e-01	-4.659704e-01	-8.074637e-01	-8.622675e-01
50%	1.402518e-01	3.178103e-02	6.183060e-01	7.240526e-01	1.824264e-03
75%	1.010572e+00	8.275838e-01	6.183060e-01	7.240526e-01	8.659160e-01
max	1.590785e+00	1.737073e+00	1.160446e+00	1.013018e+00	1.706965e+00

In [13]:
```
# разделим выборку
temp_X_train, temp_X_test, temp_y_train, temp_y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

3. Обучение моделей

3.1 Случаный лес

Попробуем случайный лес с гиперпараметром n=100 и максимальной глубиной 25.

In [14]:
```
ran_100 = RandomForestRegressor(n_estimators=100,max_depth=25)
ran_100.fit(temp_X_train, temp_y_train)
```

Out[14]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=25, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [15]:
```
print("MAE:", mean_absolute_error(temp_y_test, ran_100.predict(temp_X_test)))
print("MSE:", mean_squared_error(temp_y_test, ran_100.predict(temp_X_test)))
```

```
MAE: 4.595490006014878
MSE: 125.00996049459796
```

Попробуем случайный лес с гиперпараметром n=200 и максимальной глубиной 10.

In [17]:
```
ran_200 = RandomForestRegressor(n_estimators=200,max_depth=10)
ran_200.fit(temp_X_train, temp_y_train)
```

Out[17]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=10, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=200, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [18]:
```
print("MAE:", mean_absolute_error(temp_y_test, ran_200.predict(temp_X_test)))
print("MSE:", mean_squared_error(temp_y_test, ran_200.predict(temp_X_test)))
```

```
MAE: 8.599517955964654
MSE: 199.89646039220298
```

3.2 Градиентный бустинг

In [19]:
```
# гиперпараметр равен 100
gr_100 = GradientBoostingRegressor(n_estimators=100)
gr_100.fit(temp_X_train, temp_y_train)
```

Out[19]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

In [20]:
```
print("MAE:", mean_absolute_error(temp_y_test, gr_100.predict(temp_X_test)))
print("MSE:", mean_squared_error(temp_y_test, gr_100.predict(temp_X_test)))
```

```
MAE: 9.914969464605694
MSE: 255.68970623453973
```

In [22]:
```
# гиперпараметр равен 200
gr_200 = GradientBoostingRegressor(n_estimators=200)
gr_200.fit(temp_X_train, temp_y_train)
```

Out[22]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=200,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

In [23]:
```
print("MAE:", mean_absolute_error(temp_y_test, gr_200.predict(temp_X_test)))
print("MSE:", mean_squared_error(temp_y_test, gr_200.predict(temp_X_test)))
```

```
MAE: 8.925352011497793
MSE: 223.92218233105484
```

Сравнение качества полученных моделей

Лучшим методом стал случайный лес с гиперпараметром 100. Метрики этого метода MSE=125,0 и MAE=4,6.

Градиентный бустинг показал себя несколько хуже. При гиперпараметре n_estimators=200 метрики этого метода MSE=223,0 и MAE=4,6.